Final report of the ATLAS AOD/ESD Definition Task Force

Kétévi A. Assamagan [*], Dario Barberis [†], Davide Costanzo [‡], Edward Moyse [§], Giacomo Polesello [¶], David Quarrie [‡], David Rousseau [||](chair), RD Schaffer [||], Peter Sherwood [**]

ATLAS-SOFT-2004-006, 14 December 2004, Revision 4

Abstract

The first prototype of persistent representation of ATLAS data, as available in release 9.0.2, is described by detailing its three major components in addition to raw data: ESD, the detailed reconstruction output, AOD, the compressed form meant to be used by most analyses and the TAGS relational database entry allowing quick event preselection. The requirements and constraints which have lead to these definitions are exposed, as well as the broad lines of the computing model.

1 Introduction

ATLAS is proceeding in 2004 to its second data challenge (DC2), which emphasis is to demonstrate the validity of its computing model [1], developed to handle the very large amounts of data expected (1 PetaByte/year). The present document provides the definition of the data objects which will be produced by reconstruction and used in the final analysis stage of the simulated data produced in the framework of Data Challenge 2 (DC2).

Even though the definition of the computing model for the ATLAS experiment is still in a phase of development, it was necessary for the DC2 exercise to define in detail a fully

[*]BNL
[†]CERN / Genoa University
[‡]LBNL
[§]CERN
[¶]CERN/ INFN Pavia
[||]LAL
[**]UCL

fledged Event Data Model mapping well on this computing model, in order to be able to fully develop a realistic study. As a result of the DC2 exercise, a large user community will be exposed to this model, and will field-test it for specific use cases. The AOD/ESD definition proposed is believed to be a viable prototype, however much work is still needed to arrive at a correct compromise between size and usefulness for analysis. Experience from major experiments start up show that this is often fact reach only after the first few years of data taking.

The Event Summary Data (ESD) is meant to contain a detailed output of reconstruction. The Analysis Object Data (AOD) contains all the necessary information to perform most analyses. The TAGS allow fast selection of the events to be analysed. A target size is 500 kiloBytes per event for ESD (based on Physics TDR experience with fortran software) 100 kiloBytes per event for AOD, and 1 kiloBytes per event for the TAG.

A task force was formed in spring 2004 to gather a first viable definition of ESD, AOD and tags from the requirements expressed by the different atlas software groups, and from the feedback of prototype implementation. This document reflects the status of development of the software at the time of release 9. Identified paths for future improvements are indicated in some cases.

The first section recalls briefly the pieces of computing model where ESD/AOD/TAGS fit. Section 2 lists the major overall requirements that have been identified and the architectural elements that have been deduced. Section 3, 4 and 5 then detail the content of the ESD AOD and TAGS respectively. The level of details chosen was to list the objects with their information content, but not the C++ design nor implementation.

2 Elements of the Computing Model relevant for AOD and ESD Definition

This section presents a summary of the Tier 0 [1] model as foreseen for Data Challenge 2, (see Fig 1). It is expected that the computing model will be ammended/revisited, based on DC2 experience.

Byte stream files of event data, with contents and size chosen to resemble what is expected to come from Event Filter sub-farm output managers (SFOs), will be distributed to a farm of reconstruction processors. Files will not be delivered all at once, but rather at a rate that corresponds to expected ATLAS event rates or to a selected fraction thereof. Each reconstruction job will write ESD to its own POOL ROOT files, with the declarative

[1] The Tier 0 (at CERN) is the site where the first post-processing is done

union of those files constituting the run.

Processes to read the ESD files and write AOD (also to POOL ROOT files) will also run at the Tier 0. On the order of 10 AOD streams will be supported, with definitions and selection algorithms provided by physicists, but each event is proposed to be written to exactly one of those streams. The definition of AOD will be the same for all streams, to avoid making physics analysis of samples that cross stream boundaries needlessly painful. Further refinement of AOD streams to produce more specialised samples is expected. This secondary processing will take place at Tier 1 sites.

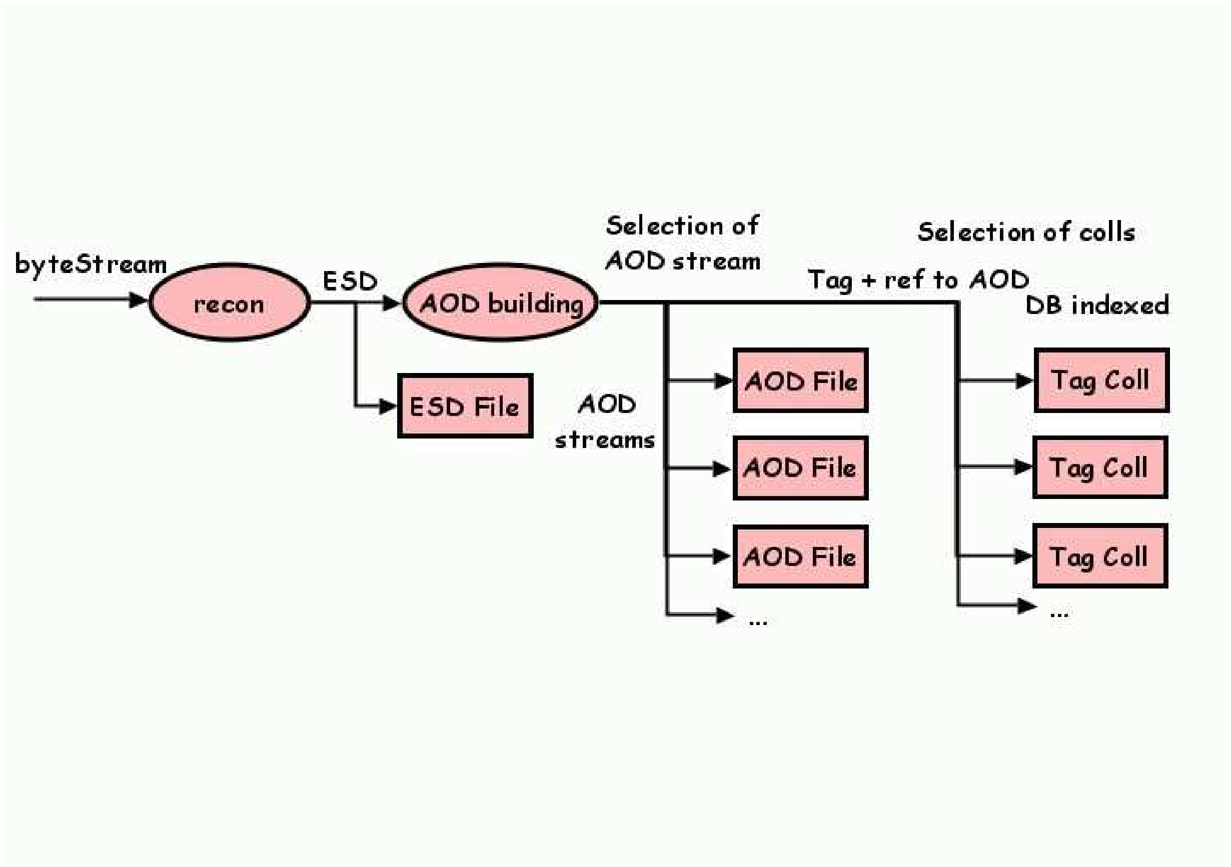

Figure 1: Data flow from raw data to the TAGS.

AOD production at the Tier 0 center ("first-pass AOD production") writes each event to exactly one stream: streams comprise a disjoint partition of the run. All streams share the same definition of AOD. The idea is not to constrain what physics working groups may ultimately choose to include or exclude as input to their analyses; indeed, it is expected that more specialized AOD production may take place routinely at Tier 1 sites. The motivation is, rather, to ensure that there is at least one version of the AOD that will support analyses that may cross stream boundaries. First-pass AOD production delivers this data product.

While it must be possible to create AOD from ESD, the RAW → ESD and ESD → AOD processes may be concatenated into combined RAW → ESD → AOD jobs for efficiency. As AOD is produced, event-level meta data (event tag data) will be written to POOL collections in local files. The number of event tag collections has not yet been decided, but there should be at least one per stream, and probably a master event tag collection for the entire run. To exercise the computing model, some collections should also cross stream boundaries. Local event tag collections produced by each job will be merged, and will reside in a relational database, with at least selected event tag attributes indexed. ESD will be distributed to at least two (but not all) Tier 1 sites, most likely in round-robin fashion by run. AOD and event tag databases will be distributed to all Tier 1 sites. Collection updating (e.g., adding todays B-physics events to B-physics events from prior runs) must be supported at sites that host such collections.

ATLAS is currently considering a model in which RAW data is also distributed to Tier 1 sites. Two alternatives are under consideration for DC2. The first is simply that RAW data for a run would be sent in byte stream format to a single Tier 1, chosen in round-robin fashion, i.e. one after the other in a cyclic manner. The second is that RAW → ESD jobs (really, RAW byte stream → ESD) may write RAW data in object format (RDOs in POOL ROOT files), in addition to writing ESD, and that this output would in turn be sent to a single Tier 1 site.

3 AOD/ESD Use Cases and Architecture

3.1 Overall requirements

The emphasis in AOD definition is to provide uniform access to the reconstructed particles (kinematics, identification variables...), with navigation pointers back to ESD objects with more detailed information.

The Event Summary Data (ESD) should contain the detailed output of reconstruction, so that refinements of combined reconstruction (e.g. particle identification, track refitting, jet calibration) can be done from the ESD, following improvements of algorithm or of calibration/alignment. This would allow very fast turn around time for the development or fine-tuning of algorithms and calibrations, and would allow these refinements to be tested on preselected data samples without large scale reprocessing. However, it will not be possible to fully redo pattern recognition, or to recalibrate all the cells, as this would require too much data to be saved. Large scale reprocessing is expected to be done from raw data.

The following use cases, constraints and requirements have been identified:

- Both AOD and ESD objects are available as objects in Athena/Gaudi framework, possibly simultaneously.
- A piece of code running on AOD should run on ESD without recompiling
- A piece of code running on Fast simulation should run on AOD (and ESD) without recompiling
- AOD and ESD should be writeable within the same job
- AOD can be produced from the ESD alone, hence
  - the ESD should have enough information to this purpose
  - there can be no direct navigation from the ESD to the AOD.
  - This may result AOD not to be bit by bit identical as in previous case, due to the compactification of the ESD objects on file. This is acceptable as long as this concerns a very small fraction of the detector resolution. But threshold effects are inevitable, e.g in some cases the number of identified particles in AOD will be different.
- it is expected that the AOD design evolves rather quickly. It should hence be possible to remake (several times) the AOD from the same ESD. This makes it difficult to share complex common base classes between AOD and ESD.
- Target size is 500 kilo byte for ESD, 50 kilo byte for AOD. However, they are expected to be larger in their first version.
- Truth information is expected to contribute significatively to the size of both AOD and ESD, hence Monte-Carlo events are expected to significatively exceed the target size.
- AOD and ESD ojects should be persistifiable with current Athena/POOL technology.
- Easy navigation from AOD to ESD, yet controllable (so that no one inadvertently accesses thousands of ESD files)
- Categorisation of an object as “AOD” or “ESD” is flexible enough so that it can be revised either in future evolution or for dedicated productions.
- A piece of code accessing ESD specific information can be easily “protected” to run on AOD without crashing

- "Partial reprocessing 1" : An algorithm that uses some ESD information can be rerun from the ESD or from the AOD in a transparent yet controllable way. (When running on AOD, navigation to ESD is used).
- "Partial reprocessing 2" : It should be possible to replace part of the input AOD or ESD by regenerating some of the objects with the same key, provided the previous object has not already been read in. This allow to run downstream algorithms and analysis without reconfiguring them. However this will invalidate objects derived from and/or pointing to the replaced objects not read in.

The target budget has been deliberately exceeded for ESD when it was felt that improvements in persistification technology could significantly reduce the size without information loss. If some short-term exercises require a size within budget, some objects can be dropped of the AOD or ESD to achieve the desired size, at a cost of a reduced scope . Apart from that, the AOD/ESD objects definition proposed, as implemented in Athena/Gaudi framework in release 9 fulfill all the requirements listed above. A more detailed list of tasks that the physicist will be able to accomplish with ESD, AOD and TAGS is given in Appendix A.

3.1.1 Framework constraint

In practice, it is expected that the list of objects to be written for the AOD/ESD is given as a list of items of an AthenaOutputStream. This list assures that converters are called for each object listed. Generic or custom written converters allow the persistification of the objects listed. Most objects use generic converters applying generic compression but special compactification of a given object can be handled by a manually written converter.

However there are two important constraints for generic converters:

- There can be only one converter for a given object type (i.e. one way of persistifying a given object type). So the same object cannot be persistify in two different ways (e.g. one detailed way and one compactified way) by its converter in the same job. [2]
- One converter can only deal with one object type. So interplay of different objects cannot be handled by the converters with some exceptions:

[2] Custom converters can still be configurable through jobOptions, provided they use tool which may have Properties. So different ways of persistifying are possible if in different jobs, hence not for AOD and ESD which should be writeable within the same job.

  - generic construct like pointers from one object to another or to an item in another container handled by the so call templated DataLink or ElementLink, which are persistifiable pointers.
  - the persistent representation of an object can be reduced if part of it can be rebuilt from the persistent representation of another object, provided this does not cause cyclical dependency.

3.2 ESD architecture

To define the ESD, the approach was to put in the ESD the complete Event Data Model detailed output of reconstruction (see [2]), dropping or compactifying the largest objects (see Fig 2). For example the Inner Detector clusters and space points are dropped[3] (hence pattern recognition cannot be redone from the ESD), but not the RIO OnTrack which allows refitting of tracks. For the calorimeter cells, a special converter is written to compactify the information (with some loss of precision).

There is the special case of TrackParticle which is the AOD representation of Track that is created already at the level of the ESD and used typically for combined reconstruction. This is because the information needed for combined reconstruction is more readily available in the TrackParticle (e.g. perigee parameters) and it simplifies building the navigation of AOD object (since a TrackParticle points to its original Track but not the opposite).

For combined reconstruction three different situations have been identified:

- The more general case is that a combined reconstruction algorithm either cannot run from ESD (not enough information) or its output is to voluminous to be an AOD object. Then this algorithm has a dedicated ESD output object (e.g. egamma for electron/photon identification), from which an AOD object with reduced information is derived (e.g Photon or Electron particle).

- In some cases, a combined reconstruction algorithm can be run from the ESD, and its output fits an AOD object. In this case, this algorithm is rather run as part of the AOD making process. This is typically the case for Btagging or Staco, one combined muon reconstruction algorithm that uses only TrackParticle.

- A last possible case is a combined reconstruction algorithm that cannot be run from ESD, and its output is an AOD object. It then can only be run in ESD making step.

[3]For sizes of the Inner Detector clusters see Table 5 in the Appendix

This is accepted only as a transitory measure, since it binds the AOD EDM to the ESD one.

To minimize the object size in the ESD (or maximize the information content), some objects are saved in the ESD file with some loss of precision. The loss of precision is by design insignificant (a small fraction of the intrinsic resolution), but sufficient for the AOD not to be bit by bit identical when built in one step (together with the ESD) or two steps (from persistified ESD). This is felt to be acceptable.

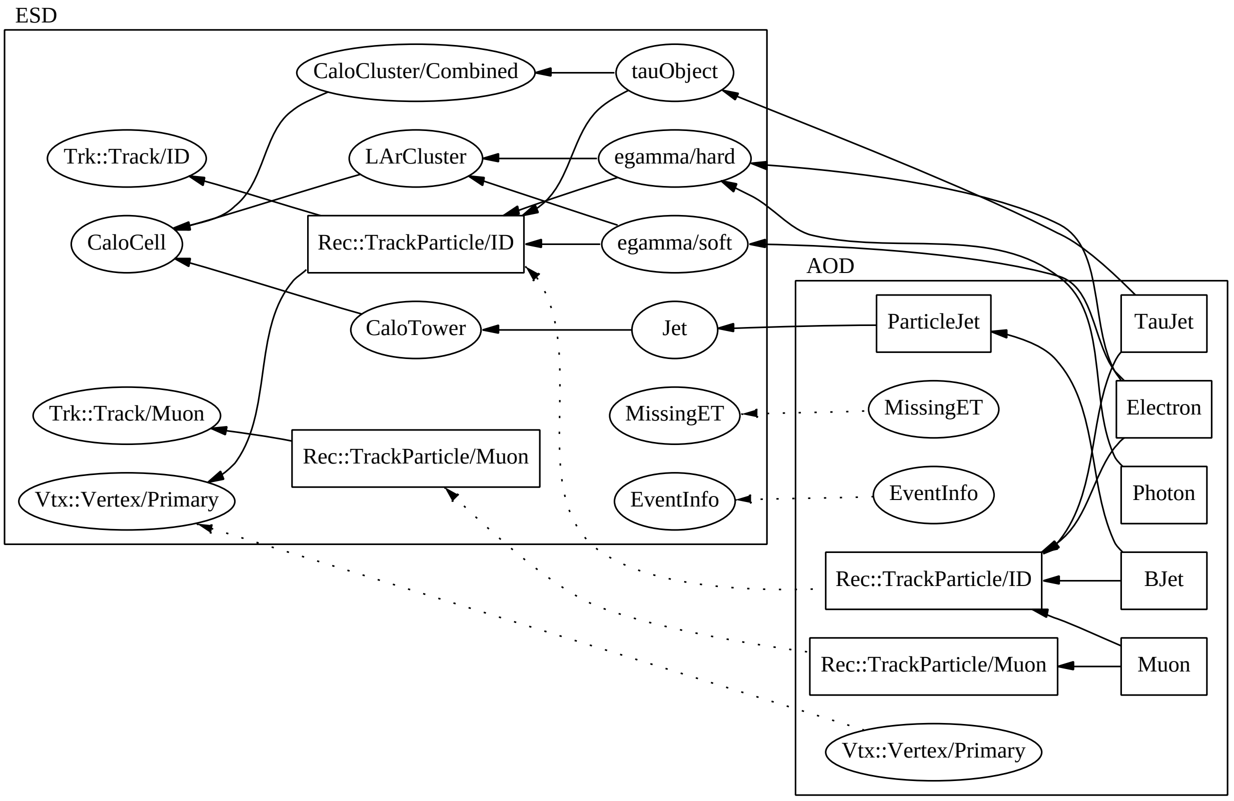

Figure 2: Main objects in ESD and AOD. Only objects collection of which are available in StoreGate are indicated. Some auxilliary objects have been omitted. Multiple instances of same objects are often omitted for clarity. Solid arrows indicate “pointed to” relationship. Dotted arrows indicate “copied from” relation ship. Object implementing the IParticle interface are indicated with square boxes. All such objects in AOD points to the primary vertex in AOD, the corresponding arrows have been omitted for clarity.

3.3 AOD architecture

The concept of slimming was explored but then ruled out: the idea was that the same object was written in the ESD then “slimmed”, (for example the list of cells of a cluster was emptied, the list of clusters on tracks was emptied) then written in AOD. The big advantage is to have one single C++ object for the same physical object, however many problems were identified with that: the difficulty to have one converter handling the AOD and ESD cases, the ambiguity to set/update pointers to these objects, the lack of flexibility of the design of the AOD, since any change in AOD has impact on ESD.

The approach which was eventually followed, was to start from the physicist needs at analysis level. Dedicated objects have been designed by the Physics and Analysis Tool group, with pointers (rather ElementLink) to detailed information in ESD objects (see Fig 2). In some cases, what is needed at analysis stage is very close to what is available from the reconstruction. In this case, the same object is available both in AOD and ESD (e.g. case of the TrackParticle).

The important use case to be able to run on ESD file with the same code as an AOD flag is accomplished by creating “on the fly” AOD objects. The probably more elegant way to have ESD and AOD object inherited from the same base class, with more details in the ESD, has been abandoned given the coupling in ESD and AOD objects design it entailed.

The baseline is that all AOD primary streams contain the same objects. This does not preclude specific AOD with specific content to be made for specific analysis, provided the impact on resource is acceptable.

4 ESD detailed content

The detailed content of ESD is given below, as in release 9. Possibilities for possible future improvements are indicated where relevant.

- EventInfo : Event ID - run/event number and time stamp, Event type - simulation, test beam and calibration/physics, and TriggerInfo containing the trigger bits from the online event.
- PileupEventInfo : For pileup events, this contains the EventInfo objects from the original events used in the pileup.
- Tracking
  - The TrackParticle (perigee parameters and hit statistics) with an ElementLink to the Trk::Track, ElementLink to the primary vertex
  - The Trk::Track will contain a list of associated RIO_OnTrack, and track parameters. The information store should be sufficient to re-extrapolate the track to any point of its trajectory, and to refit it, but not do redo

pattern-recognition (in the future one can think of saving all RIO close to a track, or all RIO on B layer, to allow redoing local pattern recognition). [4]

  - Truth: Reco track $\leftrightarrow$ true track (and RIO_OnTrack $\leftrightarrow$ true track matching), is done using std::maps, with ElementLinks as the keys.
  - Primary vertices
  - V0's (photon conversion, K^0_s and Λ decays) and hadronic interaction vertices (not available in release 9.0.0)

- Muon Spectrometer
  - The output from Muon reconstruction uses the same output classes, Trk::Track and TrackParticle. Track parameters at entrance of muon spectrometer is saved. (As for ID, in the future one can think of saving all RIO close to a track to allow for redoing local pattern recognition)

- Calorimeter
  - calorimeter cells are part of ESD, but written as a compactified object, with some reduced precision matching approximately the precision of the electronic read-out. Compactification can be turned off, which multiplies the size of the calorimeter cells by a factor six. Other possibilities that should be explored in the future, if it turns out the size of compactified cells is still too large:
    - select the cells to be saved: for example cells belonging to a cluster plus cells above threshold in term of noise
    - save P_T sum of cell of given category to allow re-weighting of missing P_T
    - save P_T sum of cell of given category for each jets to allow re-weighting

    If only some of the cells are written out, care should be taken that pointers are up to date when reading back. Easiest (but not the sole) solution is to rebuild a full CaloCellContainer where missing cells are replaced by null pointers or zero energy cells.
  - CaloCluster, parameters and ElementLink to their cells constituents.
  - combined towers with 0.1x0.1 granularity with their cell constituents, used as input to some jet clustering, are needed to keep information on which cell belong to which jet. In fact, only the tower container is saved with the minimal

[4] For release 9, the Track will be stored with all TrackParameters, but it would be possible to 'slim' the track to only have track parameters at certain places of interest (for example, perigee and where the track leaves the InnerDetector and enters the calorimeter).

information so that the towers can be rebuilt from the compactified cells, given that the cell $\leftrightarrow$ tower association is fixed once and for all.

- E/gamma
  - e/gamma : output of the different electron identification algorithm is saved in much details. Note that there is one egamma object per cluster or track. In the future, it is desirable to make a very lose pre-selection, as done for the AOD.
- Jet/Missing E_T
  - jets : Kt and cone Jet 4 momentum. Kt and cone truth jet 4 momentum, with ElementLink to their constituents.
  - tau : tauObject : there is one tauObject per combined cluster, no selection is made. Very loose cuts could be applied.
  - MissingET with different calibration scheme applied.

    MissingET needs to be computed directly from the cells not from derived objects, but the calibration of the cells depend of the particle hypothesis of nearby objects (electromagnetic or not). An issue is how to make MissingET (or jet by the way) consistent with the particle hypothesis for a given analysis. If necessary, it should be investigated to store enough information together with MissingET to recompute MissingET according to different particle hypothesis.
  - Energy flow objects. They are switched off in release 9 given that they are still experimental. However, objects (or extension of existing one) providing the best estimate of the energy flow in the event taking into account all available information are clearly needed in ESD and even in AOD.
- Trigger : A minimum requirement would be to know the trigger decisions at each level, but more information would allow trigger studies to be done from the ESD. This would consist of the following information (not available in release 9):
  - LVL1: Trigger ROIs, CTPDecision
  - LVL2: sufficient information to redo the Trigger Hypothesis algorithms
    - Track parameters: 20 words/track 5 tracks/event : 100 doubles
    - Spacepoints on track: 10 words per SP, 7 SP per track : 70 doubles
    - Calo Clusters: T2Calo output: 20 words 2 clusters/event : 40 doubles
    - EmShowerMinimal: 50 words x 2 clusters/event = 100 words
    - TriggerElements: average of 3 per event each 5 doubles + 5 ElementLink

- Event Filter: about 40 doubles, consisting of:
    - eGamma: ∼2 object per event (which include the contained Track, Cluster etc.)
    - Muon: ∼2 object per event (which include the contained Track, Cluster etc.)

- Combined muon
  - The AOD MuonParticle is used to store the output of combined muon reconstruction algorithms STACO and MUID. It has ElementLink to the combined muon trackParticle, the original muon TrackParticle and ID TrackParticle (which themselves have ElementLink to their corresponding Trk::Track). Low Pt muon identification with tile and first muon station will also be made available in the future. In the future, dedicated combined muon reconstruction objects are needed at the level of ESD, as for electron identification.

- Simulation
  - Physics event truth: there are two types of truth information, one is coming from the Generators (e.g. Pythia, Herwig, ...) and one is coming from Geant recording particles interacting within the detector. The two sets of information are linked together into a single HepMC structure, so navigation from the final state particles to the Generator truth is easiest. The full information is saved in ESD, but it has to be reduced for AOD. The Generator tree is condensed by removing all the details of the parton showering evolution. A Geant process is recorded in the tree only if the primary particle has E>1 GeV and at least one of the secondaries produced have E>300 MeV, to avoid the recording of soft processes. Both energy thresholds are user selectable.
  - Pile-up Truth: because of disk-space constraints, it is not possible to save the truth information for the secondary (pile-up) interactions. However such information will be recorded for the minimum bias events to be used for the pile-up merging. If needed the pile-up truth can be accessed by navigating from the ESD file to the minimum bias simulation files. However it should be kept in mind that since a few hundred events can be superimposed to each primary scattering, this has to be used sparingly.
  - The four-momenta of particles traversing a layer of the detector at different radii can be recorded. The particles entering the MuonSpectrometer are recorded in the present ESD prototype, this can be extended to save the list of particles entering the EM and/or hadron calorimeters if the file size allows it.

5 AOD

A first high level sketch of the analysis domain classes was proposed by the Reconstruction Task Force (see Figures 23 and 24 of the RTF report[2]), but the first real design was proposed during the UCL workshop [3]. The design and implementation have then evolved through discussions in the EDM (Event Data Model) meetings, physics Analysis tools meetings and during the software workshop at BNL. These classes serve two purposes:

- In the analysis preparation domain, they are the classes for the AOD objects, and as such, they are designed to satisfy the requirements of section 3 in addition to persistency and navigation requirements;
- In the analysis domain, they are the user analysis classes and were designed to meet further requirements of common analysis helper tools and of object associations.

The definition of these classes will evolve to reflect the exact definition of ESD/AOD. In the analysis domain, individual and physics groups may extend these classes as may be necessary in some analysis cases.

It was also noted that the AOD content could depend of the physics analysis and evolve with time. In fact, nothing precludes the different AOD streams to have different contents, however for the software release 9, it is preferred to have one single inclusive content which can satisfy all analyses.

Below is the list of the single stream AOD content for the software release 9. The quantities that cannot be calculated from the content of the AOD, i.e., without accessing the ESD, should be identified, and the impact on the analyses clearly pointed out. This exercise should be carried out for the fast simulation output objects as well. In process of defining the ESD/AOD objects with the all requirements stated herein, the event data model has gone through some changes from the software release 8.0.0 to 9.

5.1 AOD Content

- EventInfo: Event ID - run/event number and time stamp, Event type - simulation, test beam and calibration/physics, and TriggerInfo containing the trigger bits from the online event.
- Tracking
  - The TrackParticles (also in ESD). This consists of the following:

        * The Inner Detector TrackParticles
        * The MOORE TrackParticles (both at the entrance of the Muon Spectrometer and extrapolated to the primary vertex) (MuID stand-alone)
        * Muonboy TrackParticles (both at the entrance of the Muon Spectrometer and extrapolated to the primary vertex)
        * MuID TrackParticles: (MOORE, MuID standalone and MuID combined)
        * STACO TrackParticles

  In the future, we can envisage to drop the inner detector tracks and muon spectrometer tracks from AOD, at least for some trigger bits and AOD streams.

- Vertexing : primary and secondary vertices (also in ESD),
- The missing E_T object, also in ESD, as explained in the section 4.
- Trigger flags i.e the LVL2 and EF result with bits set corresponding to the trigger signatures satisfied (this is just a few words)
- The particles. As explained above, in this section 5, the base line design of the particle classes was arrived at during the UCL workshop and has done through several iterations following discussions on the EDM and physics analysis tools meetings [3]. All particles share a common design so that they can be easily used and combined. The particle candidates are created by the AOD builders, see section 5.2:
    - The Photon candidates: converted and non-converted photons.
    - The Electron candidates: hard electron;, soft electrons; and their best combination.
    - The Muon candidates: hard muons; soft muons; and their overlap is foreseen but not implemented at time of this writing.
    - The Tau Jet candidates - hadronic tau jets, one or three-prong.
    - The B-jet candidates - life time tagging and secondary vertex tagging. Other b-tagging algorithms are foreseen.
    - The Particle Jet candidates: include q/g-jets and the onces tagged as b-jets.
- Fast simulation: Particle candidates (fast photons, electrons, muons, taus, jets, and b-jets); and fast simulated missing energy, filled from Atlfast output.
- Simulation

  - A selected list of particles are saved in the AOD, by using a selection similar to what is now done with the CBNT_SpclMC. This information can be saved in a HepMC tree and event navigation can be allowed (e.g. from a Z to the decay products).
  - Monte-Carlo truth jets made with all stable particles (which are not all available at the level of AOD)

5.2 AOD Builders

AOD builders are specialised ATHENA algorithms that take the ESD (or the output of combined reconstruction) as input and produce the collections of the AOD objects of section 5.1. The codes for the AOD builders can be found in the off-line CVS repository under PhysicsAnalysis/XyzID. Each builder consists of an ATHENA algorithm and a utility or a tool specific for the construction a particular AOD object. For the software release 9, the physics analysis tools groups and b-tagging ID group have provided some implementation of the various AOD builders. Eventually, these builders should come under the control of the combined performance groups and/or the physics groups. These groups will have the responsibility of implementing the default or recommended cuts and the algorithms to build the various particles and their overlaps. The physics analysis tools group will continue to provide necessary tools. The reader may refer to the UCL workshop summary report for further details on the particle builder algorithms, sometimes refereed to as particle definition algorithms [3].

It was pondered whether AOD builders would be normally run in the ESD building process, and the AOD objects written in the ESD file, so that AOD building would be a simple filtering of objects. This was rejected since (i) AOD are streamed and not the ESD (ii) it is expected that AOD are remade several times from the same ESD, so that the normal way of working is to use the AOD files.

6 Event Tags

6.1 The Model

An event tag is an event-level meta data that allows one to make a first-pass decision about whether an event is interesting for a particular analysis or not, with an ntuple-like query interface. Tags contain pointers to events in the persistency storage. Event collections are list of pointers to events in the persistency storage, along with the event tags. The ATLAS tag databases are event collections with agreed upon tags. One would be able to query the resulting tag database to get the list of pointers that satisfy the query, and iterate over only those events. There may not be a data stream that contains only the events of interest; however, it will be straightforward to construct an event collection for just the events of interest.

Suppose you have constructed your own collection of events, say “MyFavoriteEvents”. Such a collection can be used in two ways:

- The ATHENA Event Selector will follow pointers to deliver these to your job.
- A run extraction utility will take ”MyFavoriteEvents” as input. These events can be saved in a personal files. The tools to do this are being delivered by the database group for the Data Challenge 2.

At the Tier 0 centre, the ESD and AOD of an event are constructed once. However, pointers to that event can be added to many different collections depending on the interest of physics groups. At Tier 1 centres, different groups will extract the data samples corresponding to their event collections.

6.2 Event Tag Builder Algorithms

The tag builder is an ATHENA algorithm where the initialize() method specifies the fields that go into the event tag, and the execute() method fills the tag with the quantities for each event. The database group provides the tools to add pointers to the event’s location in the persistency storage. The tag builder algorithm is similar to the process of booking and filling an NTuple, except the procedure is based upon the POOL AtributeList infrastructure. Registration streams provide a way to add a tag (the tag object is constructed by the tag builder algorithm and recorded in StoreGate) to an event collection. The tag builder algorithms can be found in the CVS repository under offline/PhysicsAnalysis/EventTag.

6.3 Event Tag Content for Release 9

It is expected that physics groups will define the tag contents for their events of interest. For release 9, a prototype tag content has been defined and used in the tag builder algorithms mentioned above:

- Number of vertices found, number of reconstructed tracks, summed jet missing E_T, missing p_x, p_y and the $sumE_T$ in the calorimeter.

- Electron: number of electrons with p_T greater than some threshold; and for each electron, signed p_T, η, ϕ, and origin vertex , and the isolation energy in some isolation cone used by the builder algorithm.
- Muon: number of muons with p_T greater than some threshold; and for each muon, signed p_T, η, ϕ, and origin vertex, and the isolation energy in some isolation cone used by the builder alogorithm.
- Photon: number of photons with energy greater than some threshold; and for each photon, the energy, η, ϕ and origin vertex.
- Tau jet: number of tau jets with p_T above some threshold; and for each tau jet, p_T, η, ϕ, the charge sign, the number of charged tracks and origin vertex.
- Jets: number of jets with p_T above some threshold; and for each jet, p_T, η, ϕ, the vertex and the probability or likelihood for b and c-jets.
- Trigger: the triggers which fired for the event and the trigger thresholds. The Central Trigger Processor (CTP) decisions and the threshold obtained from the level 1 muon region of interests (ROI) are implemented in the event tag.

7 Conclusion

A first complete prototype of ESD/AOD/Tags was described. The sizes obtained in recent release is given in Appendix B. It is expected that it evolves, (in particular in its detailed implementation which was deliberately not described), following improvements in the persistification of objects. But of foremost importance is the feedback of physicists who will start using this prototype for their own analysis.

Acknowledgements

This report is indeed the result of work and discussion of the whole ATLAS software community. In particular should be acknowledged the contribution of the architecture team and database group for the architecture, physics and analysis tool for the AOD definition, detector software, simulation and reconstruction and Event Data Model groups for the exact ESD definition and implementation.

References

[1] D. Adams *et al.*, the ATLAS Computing Model, ATL-SOFT-2004-007

[2] V. Boisvert *et al.*, Final Report of the ATLAS Reconstruction Task Force, ATL-SOFT-2003-010

[3] F. Akesson *et al.*, The UCL Summary report, http//www.usatlas.bnl.gov/PAT/

Appendix A

A Typical tasks involving TAG/AOD/ESD

Given overlaps, access to all events with a given signature involves several streams, hence whenever a physicist want to access all events of a given signature, access of AOD is necessarily done through TAG. Access to ESD is done through AOD and TAG except if one want to run on all data (direct access from TAG to ESD is also possible). A special case where this is not necessary is when dedicated ESD or AOD collections were made (extracted from data or general or dedicated MonteCarlo sample). In the following, this usage of TAG and AOD will not be mentioned anymore, i.e. a task which is annouced to be done on ESD only, will implicitly need TAG in the cases just indicated.

It should also be recalled that the back navigation feature means that if information is spread among AOD and ESD, only the AOD is used as input, back navigation allowing to reach transparently to the previous steps. For example, reaching to full detail of electron ID is done transparently by de-referencing in the electron object (in AOD) the pointer to the egamma object (in ESD) [5]. There is (currently) in the data model no back navigation to raw data. Access to RAW data in bytestream form (which is currently the default scenario) is necessarily sequential. For MonteCarlo, and also for data in some scenario, RAW data can be available in RDO format, in which case random access and back navigation is in principle possible. As part of combined reconstruction is run in ESD making step, part is done in AOD making step, the complete detailed output of reconstruction is spread between both ESD and AOD.

An algorithm can be categorised as:

- "RAW-based" algorithm if part of the information it needs is only available in RAW data
- "ESD-based" algorithm if part of the information it needs is only available in ESD data, the rest being possibly available in AOD
- "AOD-based" algorithm if all the information it needs is available in AOD data

In principle, an "AOD-based" algorithm can be run from an ESD file (or even a RAW file) provided the AOD objects needed are created on the fly by calling the algorithms needed.

[5]This is allowed only if a specific parameter of the job is set: EventSelector.BackNavigation = True

In general, system reconstruction algorithms are RAW-based, except most calorimeter algorithm (given that in the scheme proposed all cells are available in ESD). In general combined reconstruction are “ESD-based” algorithms, but some could be RAW-based, even if this is not the case today (for example if combining e.m cluster and track information requires reaching back to RIO or even RDO), some could be AOD-based, like b-tagging. “Particle Builders” that build most AOD objects are ESD-based algorithms. Table 1 list the main ESD and AOD building algorithms.

Algorithm	Input	Output object	Location
CaloCellMaker	RAW	LArCell/TileCell	ESD
LAr clustering	ESD	LArCluster	ESD
Calo clustering	ESD	CaloCluster/CaloTower	ESD
iPatrec/xKalman	RAW	Trk::Track	ESD
TrackParticle maker	ESD	Trk::TrackParticle	ESD and AOD
Primary vertex reconstruction	ESD	VxCandidate	ESD and AOD
Jet reconstruction	ESD	Jet	ESD
Moore	RAW	Trk::Track	ESD
Moore TrackParticle maker	ESD	Trk::TrackParticle	ESD and AOD
Muonboy	RAW	Trk::Track	ESD
Muonboy TrackParticle maker	ESD	Trk::TrackParticle	ESD and AOD
egamma	ESD	egammaObject	ESD
tauRec	ESD	tauObject	ESD
Energy flow object	ESD	EFlow objects	ESD
LVL1	RAW	AODLVL1	ESD and AOD
Missing ET	ESD	MissingET Object	ESD and AOD
Electron particle maker	ESD	Electron	AOD
STACO	AOD	Muon	ESD and AOD
Muid	ESD	Muon	ESD and AOD
tauJet maker	ESD	tauJet	AOD
b tagging	AOD	bJet	AOD

Table 1: Main ESD and AOD making algorithms. Output objects are indicated with their location (transient objects are omitted).

Typical tasks accomplished by physicists are listed below. In each case, it is specified wether the information is taken from TAG, AOD, ESD, with the caveat mentionned above.

- Making a preselection of events with a final state suitable for a given topology, the output of it is a new TAG collection: **TAG**. Saving the intermediate TAG collection can be skipped to run directly on the selected AOD. For example, for H$\rightarrow \gamma\gamma$ analysis : select events with at least 2 photons with sufficient Pt, with invariant mass within a window.

- detailed study of a reconstruction algorithm output : **ESD** in general
- redo part of reconstruction : **RAW**, **ESD** or **AOD**, depending wether the algorithm itself is RAW-based, ESD-based or AOD-based (e.g. reclustering calorimeter, given that cells are available).
- make an analysis : **AOD** except for very specific use cases (e.g.

  involving massive slow moving particles)
- Change identification cuts in an analysis :**AOD** (enough information is provided in AOD to tune some of the main cuts)
- make an analysis redoing part of the identification process involving at least one ESD-based algorithm : **AOD+ESD**, with back navigation. For example:
  - Refit muon with improved treatment of the overlap region
  - Use improve cell weighting in jet reconstruction
  - Improved brem recovery for electron (provided it does need RIO nor RDO)
  - Fine tuning of tracker alignment (with alignment changes small enough that redoing pattern recognition would not alter the track to cluster association)

  But as soon as a RAW based algorithm is involved, the full reconstruction need be redone (no more partial reconstruction), given that there is no back navigation to raw data.
- make an analysis redoing part of the identification process

  involving only AOD-based algorithm : **AOD**. For example:
  - tune b-tagging algorithm
  - adapt MissingET

If a preselection can be made to reduce sufficiently the event sample, it is up to the physicist to decide whether it's more optimal to make his own AOD, with objects which are normally part of the ESD.

Appendix B

B File sizes

These are the current file sizes associated with AOD/ESD, as of release 9.0.2. The detailed breakdown is only a rough estimate (due to the difficulties of measuring the sizes of persistified data in Pool). Also these measurements will quickly become outdated as the object designs and persistency mechanisms mature.

Table 2 shows the sizes of the objects present in one of the DC2 ESD files reconstructed with release 9.0.2 a non-pileup, mixed physics dataset meant to be representative of the event mixture written by Atlas [6]. There are some objects with zero size: this is a threshold effect, where the sizes of these objects are too small to be reported, as the non-zero results summed are almost equal to the complete size on disk. The muon tracks and all trigger information are missing, and there are neither reconstructed pileup, nor AOD produced from DC2 at the time of writing.

As a comparison, Table 3 shows the sizes of objects produced by running the full reconstruction (release 9.0.2) on 30 HW $\rightarrow$ bb$\mu\nu$ events both with and without pileup [7]. The pileup events have in average 200 tracks and 10 vertices per event on average, whist for the non-pileup this is 30 tracks and 1 vertex on average. Furthermore, the track size in the no pileup case is inflated (by a factor about 2) by random muon tracks, caused by missing shielding in the simulation. Muon reconstruction was not run in the pile-up case, for the same reason. McEventCollection contain both generator and Geant4 particles in the non pile-up case, only generator particles in the pile-up case.

Table 4 shows the sizes of AOD objects created from the same events as table 3.

Finally, table 5 shows the size of the Inner Detector PrepRawData (Reconstruction Input Objects) on the same data.

[6] dc2.003501.recon2.all_phys._01984.pool.root.1

[7] dc2.002055.pyt_wh120_munubb.pileup806._0001.pool.root and dc2.002055.pyt_wh120_munubb.806._0001.pool.root respectively

Object	Size/Event [KB] (w/o pileup)
`Tracks`	258
`LArClusterContainer`	31.7
`CaloCompactCellContainer`	244
`JetCollection`	38.5
`CaloClusterContainer`	15.5
`McEventCollection`	93.8
`CTP_Decision`	$\simeq$0
`CaloTowerContainer`	$\simeq$0
`DataHeader`	2.06
`EventInfo`	$\simeq$0
`LVL1_ROI`	0.037
`MissingEtCalo`	0.313
`MissingEtTruth`	0.046
`MuonContainer`	0.013
`TrackParticleContainer`	7.09
`TrackParticleTruthVector`	0.356
`TrackTruthVector`	0.349
`VxContainer`	3.78
`egammaContainer`	5.72
`tauContainer`	0.467
`TOTAL`	686

Table 2: Objects present in the DC2 ESD and their sizes on disk, averaged over 200 events from a non-pileup, mixed event dataset.

Object	No pileup Size/Event [KB]	Pileup Size/Event [KB]	Comment
`EventInfo`	≃0	4	
`DataHeader`	1	1	
`VxVertex`	4	7	
`Tracks`	631	1332	3 collections w/ pileup, 7 w/o
`TrackParticles`	12	21	1 collection w/ pileup, 4 w/o
`MuonContainer`	1	≃0	Missing from pileup
`TrackTruth`	1	≃0	
`TrackParticleTruth`	1	≃0	
`McEventCollection`	95	36	1 collection w/ pileup, 2 w/o
`JetCollection`	45	38	6 collections w/ pileup, 6 w/o
`E/Gamma`	69	15	2 collections w/ pileup, 2 w/o
`TauContainer`	≃0	1	
`LArClusterContainer`	38	96	4 collections w/ pileup, 4 w/o
`CaloCompactCells`	244	278	
`CaloClusterContainer`	15	60	2 collections w/ pileup, 2 w/o
`CaloTowerContainer`	≃0	≃0	
`MissingEtCalo`	≃0	≃0	2 collections w/ pileup, 2 w/o
`MissingEtTruth`	≃0	≃0	
`TOTAL`	1300	1900	

Table 3: ESD object sizes averaged over 30 events, from HW $\rightarrow$ bb$\mu\nu$ files both with and without pileup.

Object	Size/Event [KB] (w/o pileup)	Size/Event [KB] (w/ pileup)	Comment
`EventInfo`	$\simeq$0	4	
`DataHeader`	$\simeq$0	1	
`Primary Vertex`	1	5	
`TrackParticles`	12	19	1 collection w/ pileup, 4 w/o
`MuonContainer`	1	$\simeq$0	Missing from pileup
`Trigger Info`	$\simeq$0	$\simeq$0	Missing
`BJetContainer`	2	2	
`TauJetContainer`	1	1	
`ParticleJetContainer`	1	1	1 collection w/ pileup. 2 w/o
`PhotonContainer`	1	2	
`ElectronContainer`	2	3	
`TruthParticleContainer`	2	1	
`MissingEtCalo`	$\simeq$0	$\simeq$0	
`MissingEtTruth`	$\simeq$0	$\simeq$0	
`TOTAL`	27	47	

Table 4: Average sizes of AOD objects, on same events as Table 3.

Object	Size/event [MB]
`PixelCluster`	0.591
`SCT_Cluster`	1.44
`TRT_DriftCircle`	1.37
`TOTAL`	3.4

Table 5: Averaged sizes of Inner Detector PrepRawData (RIO) objects, for 10 events, on a HW $\rightarrow$ bb$\mu\nu$ file with pileup